Express Label EL627131377US
Date of Deposit: June 20, 2000

DETERMINING THE EQUIVALENCE OF TWO SETS OF SIMULTANEOUS LINEAR ALGEBRAIC EQUATIONS

Technical Field of the Invention

The present invention relates to a computer implementable method, and in particular, to a method and apparatus for determining whether two sets of simultaneous linear algebraic equations are equivalent.

Background Art

In many applications, the need arises to solve one or more systems of simultaneous linear algebraic equations (SLAEs) whose coefficient matrices comprise only numerical elements. Such applications include engineering and simulation computer codes. Solutions of the SLAE are typically obtained by using the well-known Gaussian elimination method. Therefore, prior methods typically would solve two such SLAE systems S_1 and S_2, and compare their solutions. However, such methods may not always work if one or both of the SLAEs are ill-conditioned and/or the numerical precision used in computations is not high enough.

Furthermore, such methods are generally not adapted to solving a set of SLAEs whose coefficient matrix elements are algebraic expressions, and for which the solution will, in general, be in algebraic form.

Disclosure of the Invention

It is an object of the present invention to provide a method of determining whether two sets of simultaneous linear algebraic equations are equivalent.

The invention provides a computer implemented method for determining the equivalence of two sets of simultaneous linear algebraic equations (SLAEs), each of the sets comprising two or more algebraic equations. The method comprising the steps of:

reducing each SLAE to a standard form; and

comparing the SLAEs to determine whether equivalence exists.

The invention further provides a computer implemented method of determining the equivalence of a first and a second set of simultaneous linear algebraic equations (SLAEs), with the method comprising the steps of:

iteratively eliminating unknowns from each of the sets of SLAEs to place each SLAE in a two-part standard form; and

forming a product of a part of one standard form equation with a part of another part of another standard form equation;

forming a product of the other part of standard form equation with the other part of another standard form equation; and

comparing the respective products for mathematical equivalence.

There is further provided a computer implemented method of determining the equivalence of a first and a second set of simultaneous linear algebraic equations, each of the equations being of a form:

$$e_{i1}x_1 + e_{i2}x_2 + e_{i3}x_3 + \ldots + e_{in}x_n = b_i$$

wherein x_j are unknowns, e_{ij} are coefficients, and b_i are quantities. The coefficients and quantities are known algebraic expressions. The method comprising the steps of:

iteratively eliminating the unknowns from each of the sets of simultaneous linear algebraic equations until each of the equations are in the form:

$$(l_{ii})_k \; x_i = (r_i)_k$$

wherein l_{ii} and r_i are algebraic expressions, and k={1;2} indicate one of the sets that the equation is derived from; and

comparing, for each of the unknowns, the products $(l_{ii})_1 * (r_i)_2$ and $(l_{ii})_2 * (r_i)_1$, wherein the first and the second set of simultaneous linear algebraic equations are equivalent if the products match for all the unknowns.

The invention further discloses a computational apparatus for determining the equivalence of a first and a second set of simultaneous linear algebraic equations, each of the equations being in the form:

$$e_{i1}x_1 + e_{i2}x_2 + e_{i3}x_3 + \ldots + e_{in}x_n = b_i$$

wherein x_j are unknowns, e_{ij} are coefficients, and b_i are quantities, the coefficients and quantities being known algebraic expressions. The apparatus comprising:

means for iteratively eliminating the unknowns from each of the sets of simultaneous linear algebraic equations until each of the equations are in the form:

$$(l_{ii})_k \ x_i = (r_i)_k$$

wherein l_{ii} and r_i are algebraic expressions, and $k=\{1;2\}$ indicate one of the sets that the equation is derived from; and

means for comparing, for each of the unknowns, the products $(l_{ii})_1 * (r_i)_2$ and $(l_{ii})_2 * (r_i)_1$, wherein the first and the second set of simultaneous linear algebraic equations are equivalent if the products match for all the unknowns.

The invention yet further discloses a computer program product carried by a storage medium for determining the equivalence of a first and a second set of simultaneous linear algebraic equations, each of the equations being of a form:

$$e_{i1}x_1 + e_{i2}x_2 + e_{i3}x_3 + \ldots + e_{in}x_n = b_i$$

wherein x_j are unknowns, e_{ij} are coefficients, and b_i are quantities, the coefficients and quantities being known algebraic expressions. The computer program product comprising:

a program element for iteratively eliminating the unknowns from each of the sets of simultaneous linear algebraic equations until each of the equations are in the form:

$$(l_{ii})_k \ x_i = (r_i)_k$$

wherein l_{ii} and r_i are algebraic expressions, and $k=\{1;2\}$ indicate one of the sets that the equation is derived from; and

a program element for comparing, for each of the unknowns, the products $(l_{ii})_1 * (r_i)_2$ and $(l_{ii})_2 * (r_i)_1$, wherein the first and the second set of simultaneous linear algebraic equations are equivalent if the products match for all the unknowns.

Preferably, the method further includes recasting the algebraic expressions into a form of one or more token pairs arranged sequentially in a string, each of the token pair comprising an operator followed by an operand; and reducing the strings in accordance with a set of predetermined simplifying rules to obtain reduced expressions. Eliminating the unknowns from each of the sets of simultaneous linear algebraic equations is performed on the reduced strings in accordance with a set of predetermined operations.

Furthermore, the simplifying rules can comprise the steps of arranging token pairs into subgroups, arranging operand tokens in an arranged subgroup in order, reducing the ordered operands by consolidating one or more constants and eliminating variables of opposite effect to form reduced subgroups, and consolidating one or more multiple instances of similar subgroups, to produce a reduced string.

Brief Description of the Drawings

A preferred embodiment of the present invention will now be described with reference to the drawings in which:

Fig. 1 is a schematic block diagram of a conventional general-purpose computer system upon which the embodiment of the invention may be practised; and

Fig. 2 is a flow diagram of a method of determining whether two sets of simultaneous linear algebraic equations are equivalent.

Detailed Description including Best Mode

Apparatus

A general-purpose computer system 100, upon which the preferred embodiment of the invention may be practised, is shown in Fig. 1. The computer system 100 will first be described, followed more particularly by a description of a method of determining whether two sets of simultaneous linear algebraic equations are equivalent.

This method may be implemented as software, such as an application program executing within the computer system 100. In particular, the steps of the method of determining whether two sets of simultaneous linear algebraic equations are equivalent, are effected by instructions in the software that are carried out by the computer system 100. The software may be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the computer system 100 from the computer readable medium, and then executed by the computer system 100. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the

computer preferably effects an advantageous apparatus for determining whether two sets of simultaneous linear algebraic equations are equivalent, in accordance with the embodiments of the invention.

The computer system 100 comprises a computer module 101, input devices such as a keyboard 102 and mouse 103, and output devices including a printer 115 and a display device 114. The computer module 101 typically includes at least one processor unit 105, a memory unit 106, for example formed from semiconductor random access memory (RAM) and read only memory (ROM), input/output (I/O) interfaces including a video interface 107, an I/O interface for the printer device 115 and an I/O interface 113 for the keyboard 102 and mouse 103. A storage device 109 is provided and typically includes a hard disk drive 110 and a floppy disk drive 111. A CD-ROM drive (not illustrated) may be provided as a non-volatile source of data. The components 105 to 113 of the computer module 101, typically communicate via an interconnected bus 104 and in a manner which results in a conventional mode of operation of the computer system 100 known to those in the relevant art.

Typically, the application program of the preferred embodiment is resident on the hard disk drive 110, and read and controlled in its execution by the processor 105. Intermediate storage of the program may be accomplished using the semiconductor memory 106, possibly in concert with the hard disk drive 110. In some instances, the application program may be supplied to the user encoded on a CD-ROM or floppy disk and read via a CD-ROM drive (not illustrated) or floppy disk drive 111, or alternatively may be read by the user from the network (not illustrated) via the modem device (not illustrated). Still further, the software can also be loaded into the computer system 100 from other computer readable medium including magnetic tape, a ROM or integrated circuit, a magneto-optical disk, a radio or infra-red transmission channel between the computer module 101 and another device, a computer readable card such as a PCMCIA card, and the Internet and Intranets including e-mail transmissions and information recorded on websites and the like. The foregoing is merely exemplary of relevant computer readable mediums. Other computer readable mediums may be practiced without departing from the scope and spirit of the invention.

Having described the hardware environment of the invention, the method of determining whether two sets of simultaneous linear algebraic equations are equivalent will now be described.

Broad Outline of Method

Let S represent a system of simultaneous linear algebraic equations (SLAEs) as is given by the following:

$$e_{11}x_1 + e_{12}x_2 + e_{13}x_3 + \ldots + e_{1n}x_n = b_1$$
$$e_{21}x_1 + e_{22}x_2 + e_{23}x_3 + \ldots + e_{2n}x_n = b_2$$
$$\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots$$
$$e_{n1}x_1 + e_{n2}x_2 + e_{n3}x_3 + \ldots + e_{nn}x_n = b_n$$

where n-unknowns $\{x_1, x_2, x_3, \ldots, x_n\}$ are related by n equations, and coefficients e_{ij} (with $i=1,2,\ldots,n$ and $j= 1,2,\ldots,n$) are known algebraic expressions, as are the right-hand side quantities b_i, $i = 1,2,\ldots,n$.

The method of determining whether two such systems S_1 and S_2 are equivalent - that is, their respective solutions are identical to each other - broadly has two parts, namely:

(1) The reduction of each system of SLAEs S into a standard form of the type

$$l_{11}x_1 = r_1$$
$$l_{22}x_2 = r_2$$
$$l_{33}x_3 = r_3$$
$$\ldots \quad \ldots$$
$$l_{nn}x_n = r_n$$

where l_{ii} and r_i are algebraic expressions; and

(2) Comparison of two sets of SLAEs in their standard form.

It is assumed that the coefficients e_{ij} and the quantities b_i of the SLAEs S_1 and S_2 have no division operators. Undesirable division operators can be eliminated from the SLAEs S_1 and S_2 by multiplying the affected equations by appropriate factors. This is

done to reduce the complexity of handling operands associated with the division operator, which is not a commutative operator.

Reduced Expression

The coefficients e_{ij} and the quantities b_i may be written as expressions, wherein the terms in the coefficients e_{ij} and the quantities b_i may include constants and variables. In the preferred embodiment, to facilitate comparisons between two expressions, the concept of a reduced form of an expression, as described below, has been used. The reduced expression is the canonical form to which expressions are converted.

It is *apriori* assumed that the expression to be converted is syntactically correct and does not contain any blanks. In the preferred embodiment, variables are limited in their construction to lower-case alphabets, underscore character, and digits, except that a variable may not start with a digit or end with an underscore. If these construction rules are not met, then the affected variables may be mapped (aliased) to alternative, but distinct, variables obeying the construction rules, and these new variables used instead.

A convention adopted for the present embodiment is that variables in the coefficients e_{ij} and the quantities b_i raised to a positive integer power are written out as multiplications of the variables. Thus, for example:

$\mathtt{a}^{\mathtt{n}}$ becomes `a*a*...*a`, where `a` appears `n` times in the product.

To convert a given expression into a reduced expression, the expression firstly is put in the following form:

<unitary operator><operand><operator><operand>.....<operator><operand>

where the unitary operator is either + (plus) or – (minus), and each operator is one of + (plus), - (minus), or * (multiplication). In the event that an expression does not commence with a unitary operator, a unitary operator + (plus) is inserted at the start of the expression. For example:

`a+b*c-d` becomes `+a+b*c-d`

Note, in particular, the absence of brackets. Brackets, if present in the expression, must be removed by carrying out the necessary operations needed to remove

them, such as multiplying two parenthesized factors, discarding superfluous brackets, etc. to bring a given expression into the above form.

Next, all + (plus) operators are substituted with the string +1* so that + becomes +1*. Similarly, all - (minus) operators are substituted with the string -1* so that - becomes -1*. Thus, for example:

+a becomes +1*a

and

-a*b becomes -1*a*b

Finally, the operands, which are constants (including the '1's introduced in the previous step) are converted into an e-format as follows:

".[unsigned number]e[e-sign][unsigned exponent]"

where: [unsigned number] is a *n*-digited number comprising only digits and *n* is a prefixed integer greater than 0;
[e-sign] is the sign of the exponent and is one of > for plus or < for minus; and
[unsigned exponent] is a *m*-digited number comprising only digits and *m* is a prefixed integer greater than 0.

Thus, for example:

$25 = 0.25*10^{2}$ becomes .250000e>02

and

$0.025 = 0.25*10^{-1}$ becomes .250000e<01

where it is assumed *n*=6 and *m*=2. It is noted that any constant will be represented by a string of constant length *m*+*n*+3 characters in the e-format. Here e[e-sign][unsigned exponent] represents the quantity 10 raised to the power [e-sign][unsigned exponent], which must be multiplied to the number represented by .[unsigned number] to get the actual constant.

Now, the expression will contain at least one operand which is a constant. Each expression will have one or more terms, where each term has the following form:

<unitary operator><operand><*><operand>.....<*><operand>

where the unitary operator is either + (plus) or – (minus), and between two consecutive operands is the multiplication operator `*`. After the terms are identified, the [e-sign] of each constant is restored from < or > to – or + respectively.

In each term the operands are sorted (rearranged) in ascending order according to their ASCII (American Standard Code for Information Interchange) value. This does not affect the term since the multiplication operator is a commutative operator, so the exchange of operands is completely permissible. The operands, which are constants, will all bunch up at the beginning of the terms where they can be easily identified and replaced by a single constant. Thus, for example:

```
+.100000e+01*a*b*.500000e+00
```

after arranging the operands in ascending order becomes

```
+.100000e+01*.500000e+00*a*b
```

and after consolidating the constants the term becomes

```
+.500000e+00*a*b
```

At this stage a term will have the following form:

<unitary operator><constant><*><operand>.....<*><operand>

where each operand is a variable, whose ASCII value is not lower than that of its preceding operand, if any. This is the reduced form of a term. In the reduced form, the non-constant part of a term is called a variable-group. For example, if the term in the reduced form is `"+.250000e+01*a*a*b"`, then its variable-group is `"*a*a*b"`.

In an expression, all those terms whose variable-groups match, are combined by modifying the constant in one of the terms, and eliminating all other terms with identical variable-group.

Finally, the reduced terms in the expression are rearranged in an ascending order according to the ASCII value of their respective variable-group. In this final form, the expression is said to be in its reduced form. Note, in particular, that no two terms in a reduced expression will have the same variable-group.

Method of Determining Equivalence

Referring to Fig. 2, a method 200 of determining whether two such systems S_1 and S_2 are equivalent is shown. Starting in step 240, all the coefficients e_{ij} and the quantities b_i are converted into their respective reduced form (as discussed above).

In steps 250 to 280, the Gaussian elimination and back substitution method (adapted to avoid divisions) is used to bring the SLAEs S_1 and S_2 into a standard form.

In step 250 a counter k is set to 1. Step 252 follows, where the variable x_k is eliminated from the j-th equations, j = (k+1),...,n, to get a k'th derived system. In particular, with counter k equal to 1, the variable x_1 is eliminated from the j-th equations, $j = 2,3,...,n$, to get a first derived system defined as:

$$\begin{aligned} e_{11}x_1 + e_{12}x_2 + e_{13}x_3 + \ldots + e_{1n}x_n &= b_1 \\ {}^1e_{22}x_2 + {}^1e_{23}x_3 + \ldots + {}^1e_{2n}x_n &= {}^1b_2 \\ \ldots \quad \ldots \quad \ldots \quad \ldots & \quad \ldots \\ {}^1e_{n2}x_2 + {}^1e_{n3}x_3 + \ldots + {}^1e_{nn}x_n &= {}^1b_n \end{aligned}$$

where the new coefficients ${}^1e_{jk}$ of the first derived system are given by:

$${}^1e_{jk} = e_{jk}\,e_{11} - e_{1k}\,e_{j1}; \text{ and}$$

$${}^1b_j = b_j\,e_{11} - b_1\,e_{j1}, \qquad \text{for } (j,k) = 2,\ldots,n.$$

In a case where the coefficient $e_{11} = 0$, then the first equation of the system S is interchanged with any other equation m of the system S for which its coefficient e_{1m} is non-zero. If no such equation m can be found, then the SLAEs are singular, and the method 200, and in particular step 252, is interrupted by following the line 262 to step 270, where the method 200 is terminated with an appropriate error message.

In step 260 it is determined whether the counter k is equal to n-1, where n is the number of unknowns. If this is not so, a sub-system is defined in step 255 from the k'th derived system. For example, with counter k equal to 1, the sub-system derived from the first derived system is as follows:

$$\begin{array}{l} {}^{1}e_{22}\, x_2 + {}^{1}e_{23}\, x_3 + \ldots + {}^{1}e_{2n}\, x_n = {}^{1}b_2 \\ \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\ {}^{1}e_{n2}\, x_2 + {}^{1}e_{n3}\, x_3 + \ldots + {}^{1}e_{nn}\, x_n = {}^{1}b_n \end{array}$$

This sub-system is a set of (n - 1) SLAEs, having (n - 1) unknowns $\{x_2, x_3, \ldots, x_n\}$. After incrementing the counter k in step 258, the steps of reduction 252 to 260 are now repeated on the sub-systems, until the system S is reduced to a (n -1)-th derived system as follows:

$$\begin{array}{l} e_{11}\, x_1 + e_{12}\, x_2 + e_{13}\, x_3 + \ldots + e_{1n}\, x_n = b_1 \\ \qquad {}^{1}e_{22}\, x_2 + {}^{1}e_{23}\, x_3 + \ldots + {}^{1}e_{2n}\, x_n = {}^{1}b_2 \\ \qquad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\ \qquad\qquad\qquad {}^{n-1}e_{nn}\, x_n = {}^{n-1}b_n \end{array}$$

wherein the diagonal coefficients ${}^{j-1}e_{jj}$, j=1,...,n, are all nonzero, and where

$${}^{l}e_{jk} = {}^{l-1}e_{jk}\, {}^{l-1}e_{ll} - {}^{l-1}e_{lk}\, {}^{l-1}e_{jl},$$

$${}^{l}b_j = {}^{l-1}b_j\, {}^{l-1}e_{ll} - {}^{l-1}b_l\, {}^{l-1}e_{jl}, \qquad \text{for } l = 1,\ldots,n-1;\ (j,k) = l+1,\ldots,n,$$

and

$${}^{0}e_{jk} = e_{jk}.$$

This completes the Gaussian elimination phase of the process. Note the absence of any division in the entire process. The counter k is now equal to n-1 and the method therefore continues to step 280 where back substitution is performed, again without any division. Therefore, instead of calculating the unknown x_i, the product $l_{ii}x_i$ is calculated, where each of the n unknowns x_i is expressed in the form of a ratio $x_I = r_i/l_{ii}$ with r_i a numerator and l_{ii} a denominator. With $i = n$, we have,

$${}^{n-1}e_{nn}x_n = {}^{n-1}b_n$$

so that

$$l_{nn} = {}^{n-1}e_{nn} \text{ and } r_n = {}^{n-1}b_n.$$

For $i = n - 1$, the $(n - 1)$-th equation is multilpied by the denominator l_{nn} to obtain

$$l_{nn}\ {}^{n-2}e_{n-1,n-1}\, x_{n-1} + l_{nn}\ {}^{n-2}e_{n-1,n}\, x_n = {}^{n-2}b_{n-1}\, l_{nn}$$

or

$$l_{nn}\ {}^{n-2}e_{n-1,n-1}\, x_{n-1} = {}^{n-2}b_{n-1}\, l_{nn} - {}^{n-2}e_{n-1,n}\, r_n$$

so that

$$l_{n-1,n-1} = l_{nn}\ {}^{n-2}e_{n-1,n-1} \text{ and } r_{n-1} = {}^{n-2}b_{n-1}\, l_{nn} - {}^{n-2}e_{n-1,n}\, r_n$$

For $i = n - 2$, we multiply the $(n - 2)$-th equation by the denominator $l_{n-1,n-1}$ and obtain

$$l_{n-1,n-1}\ {}^{n-3}e_{n-2}\, x_{n-2} + l_{n-1,n-1}\ {}^{n-3}e_{n-2,n-1}\, x_{n-1} + l_{n-1,n-1}\ {}^{n-3}e_{n-2,n}\, x_n = {}^{n-3}b_{n-2}\, l_{n-1,n-1}$$

or

$$l_{n-1,n-1}\ {}^{n-3}e_{n-2,n-2}\, x_{n-2} = {}^{n-3}b_{n-2}\, l_{n-1,n-1} - {}^{n-2}e_{n-1,n-1}\ {}^{n-3}e_{n-2,n}\, r_n - {}^{n-3}e_{n-2,n-1}\, r_{n-1}$$

so that

$$l_{n-2,n-2} = l_{n-1,n-1}\ {}^{n-3}e_{n-2,n-2} \text{ and } r_{n-2} = {}^{n-3}b_{n-2}\, l_{n-1,n-1} - {}^{n-2}e_{n-1,n-1}\ {}^{n-3}e_{n-2,n}\, r_n - {}^{n-3}e_{n-2,n-1}\, r_{n-1}$$

It can be shown that for any $i = 1,2,\ldots,n - 1$, the result will be

$$l_{ii} = l_{i+1,i+1}\ {}^{i-1}e_{ii} \text{ and } r_i = {}^{i-1}b_i\, l_{i+1,i+1} - R_{in}\, r_n - R_{i,n-1}\, r_{n-1} - \ldots - R_{i,i+1}\, r_{i+1}$$

with

$$l_{nn} = {}^{n-1}e_{nn} \text{ and } r_n = {}^{n-1}b_n$$

where

$$R_{ij} = (l_{i+1,i+1}/l_{jj})\ {}^{i-1}e_{ij} \qquad \text{for } j = n,\ldots,(i+1) \text{ and } i = 1,2,\ldots,n - 1.$$

Note that since l_{jj} is a factor of $l_{i+1,i+1}$, R_{ij} will be free of any divisions. However, it is noted that there is no step in the back substitution step 280 where factors common to l_{ii} and r_i have been eliminated.

After completing steps 240 to 280 for each of the two SLAEs systems S_1 and S_2, string arrays $(l_{ii})_1$ and $(r_i)_1$ for system S_1 and $(l_{ii})_2$ and $(r_i)_2$ for system S_2 have been

produced. In principle, to show that the solutions of the two systems S_1 and S_2 are equivalent, it would suffice if their respective string arrays l_{ii} and r_i where shown to match. However, this can not always be done, since it is generally not possible to eliminate their common factors completely by presently known methods. It must therefore be assumed that there may be uneliminated common factors present. However, it is clear that mathematically

$$(l_{ii}/r_i)_1 = (l_{ii}/r_i)_2$$

or equivalently,

$$(l_{ii})_1 * (r_i)_2 = (l_{ii})_2 * (r_i)_1$$

in which form a comparison may be performed. Therefore, step 290 calculates expressions $(l_{ii})_1 * (r_i)_2$ and $(l_{ii})_2 * (r_i)_1$ for each $i = 1,..,n$. If all the expressions $(l_{ii})_1 * (r_i)_2$ and $(l_{ii})_2 * (r_i)_1$ have been consistently reduced to their reduced form, then a step 300 performs a simple string comparison of $(l_{ii})_1 * (r_i)_2$ with $(l_{ii})_2 * (r_i)_1$. A decision step 310 determines whether matches were found for all $i = 1,..,n$. If the answer is Yes, then equivalence of systems S_1 and S_2 is reported in step 312. Alternatively, non-equivalence is reported in step 315.

Example

An example of performing the method 200 to determine whether two systems S_1 and S_2 are equivalent, will now be described. C and C++ programing language notations will be used. In this notation, the coefficients e_{ij} and the quantities b_i are denoted as e[i-1][j-1] and b[i-1] respectively.

To understand the example given below, reference to the following pseudo-code fragment will be helpful. The variables `e[][]` and `b[]` are assumed to have the datatype algebraic "expression", which *inter alia* will implement the operators '+' (plus), '-' (minus), and '*' (multiplication) operators on such expressions. The class "expression" also has a method which can convert an algebraic expression into its reduced form.

```
// Gaussian elimination
// e[][] and b[] are of type Expression.

for (i = 0; i < n-1; i++) { // Index for the derived system.
```

```
    // --- Comment 1 ---
    // If e[i][i] = 0, exchange this row with another below it (say
    // the k-th row, k > i) such that e[i][k] != 0. If no such k is
    // found, exit with the message that the matrix e is singular.
    // The code to do this is not shown here.

    for (j = i+1; j < n; j++) {
      for (k = i+1; k < n; k++) {
         // Multiply i-th row with e[j][i].
         // Multiply j-th row with e[i][i].
         // Subtract i-th row from j-th row.
         e[j][k] = e[j][k]*e[i][i] - e[i][k]*e[j][i];
      }
      b[j] = b[j]*e[i][i] - b[i]*e[j][i];
    }

    // Zero lower triangle coefficients
    for (k = 0; k < i; k++) e[i][k] = "0";
  }
  // Back-substitute.
  i = n;

  // At the end of the following while loop, e[i-1][i-1] will
  // contain l_ii
  // and b[i-1] will contain r_i. The solution will be x_i = l_ii/ r_i.

  while (i--) {
    j = n;
    while (j--) b[j] = b[j]*e[i][i] - b[i]*e[j][i];
    for(k = 0; k < n; k++) {
      for (j = k; j < n; j++) {
        e[k][j] *= e[i][i];
      }
    }
  }
```

Now, let system S_1 be the set of equations:

$$ax_1 + x_2 + x_3 = a + 2$$

$$x_1 + x_2 + x_3 = 3$$

$$x_1 + x_2 - x_3 = 1$$

and let system S_2 be the set of equations

$$ax_1 + 2x_2 = a + 2$$

$$2x_1 + 2x_2 = 4$$

$$x_2 - x_3 = 0$$

That is, each set consists of three equations.

Considering system S_1 first, the coefficients e_{ij} and the quantities b_i may be written as follows:

e[0][0] = a
e[0][1] = 1
e[0][2] = 1
b[0] = a+2
e[1][0] = 1
e[1][1] = 1
e[1][2] = 1
b[1] = 3
e[2][0] = 1
e[2][1] = 1
e[2][2] = -1
b[2] = 1

Performing step 240 in system S_l, all the terms in the coefficients e_{ij} and the quantities b_i are converted by the computer program performing method 200 into the reduced form, with the text variable to which a pseudocode variable refers to at different stages of computation noted on the right hand side, as follows:

Reduced Form	*Variables*
e[0][0] = +.10000e+01*a	${}^{0}e_{11} = e_{11}$
e[0][1] = +.10000e+01	${}^{0}e_{12} = e_{12}$
e[0][2] = +.10000e+01	${}^{0}e_{13} = e_{13}$
b[0] = +.10000e+01*a+.20000e+01	${}^{0}b_1 = b_1$
e[1][0] = +.10000e+01	${}^{0}e_{21} = e_{21}$
e[1][1] = +.10000e+01	${}^{0}e_{22} = e_{22}$
e[1][2] = +.10000e+01	${}^{0}e_{23} = e_{23}$
b[1] = +.30000e+01	${}^{0}b_2 = b_2$
e[2][0] = +.10000e+01	${}^{0}e_{31} = e_{31}$

e[2][1] = +.10000e+01	${}^{0}e_{32} = e_{32}$
e[2][2] = -.10000e+01	${}^{0}e_{33} = e_{33}$
b[2] = +.10000e+01	${}^{0}b_3 = b_3$

With counter k set to 1 in step 250, a first derived system is found by performing step 252, thereby eliminating the variable x_1 from equations 2 and 3. The coefficients ${}^{1}e_{ij}$ and the quantities ${}^{1}b_i$ of the first derived system are as follows:

Reduced Form	*Variables*
e[0][0] = +.10000e+01*a	${}^{0}e_{11}$
e[0][1] = +.10000e+01	${}^{0}e_{12}$
e[0][2] = +.10000e+01	${}^{0}e_{13}$
b[0] = +.10000e+01*a+.20000e+01	${}^{0}b_1$
e[1][0] = +.00000e+00	${}^{1}e_{21}$
e[1][1] = -.10000e+01+.10000e+01*a	${}^{1}e_{22}$
e[1][2] = -.10000e+01+.10000e+01*a	${}^{1}e_{23}$
b[1] = -.20000e+01+.20000e+01*a	${}^{1}b_2$
e[2][0] = +.00000e+00	${}^{1}e_{31}$
e[2][1] = -.10000e+01+.10000e+01*a	${}^{1}e_{32}$
e[2][2] = -.10000e+01-.10000e+01*a	${}^{1}e_{33}$
b[2] = -.20000e+01	${}^{1}b_3$

The above first derived system for system S_1, when written in normal algebraic form, appears as:

$$ax_1 + x_2 + x_3 = a + 2$$
$$(a - 1)x_2 + (a - 1)x_3 = 2(a-1)$$
$$(a - 1)x_2 - (a + 1)x_3 = -2$$

By repeating steps 250 to 260, the method 200 calculates the second derived system for system S_1 as follows:

Reduced Form	*Variables*
e[0][0] = +.10000e+01*a	${}^{0}e_{11}$
e[0][1] = +.10000e+01	${}^{0}e_{12}$
e[0][2] = +.10000e+01	${}^{0}e_{13}$
b[0] = +.10000e+01*a+.20000e+01	${}^{0}b_{1}$
e[1][0] = +.00000e+00	${}^{1}e_{21}$
e[1][1] = -.10000e+01+.10000e+01*a	${}^{1}e_{22}$
e[1][2] = -.10000e+01+.10000e+01*a	${}^{1}e_{23}$
b[1] = -.20000e+01+.20000e+01*a	${}^{1}b_{2}$
e[2][0] = +.00000e+00	${}^{2}e_{31}$
e[2][1] = +.00000e+00	${}^{2}e_{32}$
e[2][2] = +.20000e+01*a-.20000e+01*a*a	${}^{2}e_{33} = l_{33}$
b[2] = +2.0000e+00*a-2.0000e+00*a*a	${}^{2}b_{3} = r_{3}$

or alternatively

$$ax_1 + x_2 + x_3 = a + 2$$
$$(a-1)x_2 + (a-1)x_3 = 2(a-1)$$
$$-2a\,(a-1)x_3 = -2a(a-1)$$

Performing the back substitution step 280 the numerators r_i and the denominators l_{ii} can be found. In particular, from the last equation of the second derived system the numerator r_3 and the denominator l_{33} are as follows:

$l_{33} = -2a\,(a-1)$ and $r_3 = -2a\,(a-1)$.

Substituting numerator r_3 and denominator l_{33} into the second equation, we get:

Reduced Form	*Variables*
e[1][1] = -.20000e+01*a+.40000e+01*a*a-.20000e+01*a*a*a	l_{22}
b[1] = -.20000e+01*a+.40000e+01*a*a-.20000e+01*a*a*a	r_{2}

or

$l_{22} = -2a\,(1-2a+a^2)$ and $r_2 = -2a\,(1-2a+a^2)$.

In the final back substitution we get

Reduced Form	*Variables*
e[0][0] = -.40000e+01*a*a*a+.12000e+02*a*a*a*a-.12000e+02*a*a*a*a*a +.40000e+01*a*a*a*a*a*a	l_{11}
b[0] = -.40000e+01*a*a*a+.12000e+02*a*a*a*a-.12000e+02*a*a*a*a*a +.40000e+01*a*a*a*a*a*a	r_1

producing thereby

$$l_{11} = -4a^3 (1 - 3a + 3a^2 - a^3) \text{ and } r_1 = -4a^3 (1 - 3a + 3a^2 - a^3).$$

In a similar manner, the first derived system of system S_2 may be written as follows:

Reduced Form	*Variables*
e[0][0] = +.10000e+01*a	${}^0e_{11}$
e[0][1] = +.20000e+01	${}^0e_{12}$
e[0][2] = +.00000e+00	${}^0e_{13}$
b[0] = +.10000e+01*a+.20000e+01	${}^0b_1$
e[1][0] = +.00000e+00	${}^1e_{21}$
e[1][1] = -.40000e+01+.20000e+01*a	${}^1e_{22}$
e[1][2] = +.00000e+00	${}^1e_{23}$
b[1] = -.40000e+01+.20000e+01*a	${}^1b_2$
e[2][0] = +.00000e+00	${}^1e_{31}$
e[2][1] = +.10000e+01*a	${}^1e_{32}$
e[2][2] = -.10000e+01*a	${}^1e_{33}$
b[2] = +.00000e+00	${}^1b_3$

or

$$ax_1 + 2x_2 = a + 2$$
$$2 (a - 2)x_2 = 2 (a - 2)$$
$$ax_2 - ax_3 = 0$$

The second derived system for system S_2 is as follows:

Reduced Form	*Variables*
e[0][0] = +.10000e+01*a	${}^{0}e_{11}$
e[0][1] = +.20000e+01	${}^{0}e_{12}$
e[0][2] = +.00000e+00	${}^{0}e_{13}$
b[0] = +.10000e+01*a+.20000e+01	${}^{0}b_{1}$
e[1][0] = +.00000e+00	${}^{1}e_{21}$
e[1][1] = -.40000e+01+.20000e+01*a	${}^{1}e_{22}$
e[1][2] = +.00000e+00	${}^{1}e_{23}$
b[1] = -.40000e+01+.20000e+01*a	${}^{1}b_{2}$
e[2][0] = +.00000e+00	${}^{2}e_{31}$
e[2][1] = +.10000e+01*a	${}^{2}e_{32}$
e[2][2] = +.40000e+01*a-.20000e+01*a*a	${}^{2}e_{33} = l_{33}$
b[2] = +.40000e+01*a-.20000e+01*a*a	${}^{2}b_{3} = r_{3}$

or

$$ax_1 + 2x_2 = a + 2$$
$$2(a-2)x_2 = 2(a-2)$$
$$2a\,(2-a)x_3 = 2a\,(2-a)$$

Again performing the back substitution step 280 with system S_2 the numerators r_i and the denominators l_{ii} can be found. The numerator r_3 and the denominator l_{33} are as follows:

$$l_{33} = 2a\,(2-a) \text{ and } r_3 = 2a\,(2-a).$$

Substituting numerator r_3 and denominator l_{33} into the second equation, we get:

Reduced Form	*Variables*
e[1][1] = -.16000e+02*a+.16000e+02*a*a-.40000e+01*a*a*a	l_{22}
b[1] = -.16000e+02*a+.16000e+02*a*a-.40000e+01*a*a*a	r_{2}

or

$$l_{22} = -4a\,(4 - 4a + a^2) \text{ and } r_2 = -4a\,(4 - 4a + a^2).$$

In the final back substitution we get

Reduced Form	*Variables*
e[0][0] = -.64000e+02*a*a*a+.96000e+02*a*a*a*a-.48000e+02*a*a*a*a*a +.80000e+01*a*a*a*a*a*a	l_{11}
b[0] = -.64000e+02*a*a*a+.96000e+02*a*a*a*a-.48000e+02*a*a*a*a*a +.80000e+01*a*a*a*a*a*a	r_1

or

$$l_{11} = -8a^3 (8 - 12a + 6a^2 - a^3) \text{ and } r_1 = -8a^3 (8 - 12a + 6a^2 - a^3).$$

Performing step 290, the expressions $(l_{ii})_1 * (r_i)_2$ and $(l_{ii})_2 * (r_i)_1$ are calculated and reduced to their reduced forms. For example, calculating $(l_{22})_1 * (r_2)_2$ gives the following:

$(l_{22})_1 * (r_2)_2 =$ (-.20000e+01*a+.40000e+01*a*a-.20000e+01*a*a*a) *
(-.16000e+02*a+.16000e+02*a*a-.40000e+01*a*a*a)

= +.32000e+02*a*a-.32000e+02*a*a*a+.80000e+01*a*a*a*a
-.64000e+02*a*a*a+.64000e+02*a*a*a*a-.16000e+02*a*a*a*a*a
+.32000e+02*a*a*a*a-.32000e+02*a*a*a*a*a+.80000e+01*a*a*a*a*a*a

= +.32000e+02*a*a-.96000e+02*a*a*a+.10400e+03*a*a*a*a
-.48000e+02*a*a*a*a*a+.80000e+01*a*a*a*a*a*a

Similarly, calculating $(l_{22})_2 * (r_2)_1$ gives the following:

$(l_{22})_2 * (r_2)_1 =$ (-.16000e+02*a+.16000e+02*a*a-.40000e+01*a*a*a) *
(-.20000e+01*a+.40000e+01*a*a-.20000e+01*a*a*a)

= +.32000e+02*a*a-.64000e+02*a*a*a+.32000e+02*a*a*a*a
-.32000e+02*a*a*a+.64000e+02*a*a*a*a-.32000e+02*a*a*a*a*a
+.80000e+01*a*a*a*a-.16000e+02*a*a*a*a*a+.80000e+01*a*a*a*a*a*a

= +.32000e+02*a*a-.96000e+02*a*a*a+.10400e+03*a*a*a*a
-.48000e+02*a*a*a*a*a+.80000e+01*a*a*a*a*a*a

Step 290 similarly calculates the expressions $(l_{ii})_1 * (r_i)_2$ and $(l_{ii})_2 * (r_i)_1$ for i=1 and i=3. A simple string comparison of $(l_{22})_1 * (r_2)_2$ with $(l_{22})_2 * (r_2)_1$, performed in step 300, shows that these expressions match. By repeating the comparison of $(l_{ii})_1 * (r_i)_2$ with $(l_{ii})_2 * (r_i)_1$ for i=1 and i=3, and finding that the expressions match for each i = 1, 2 and 3, it can be shown that system S_1 is equivalent to system S_2.

Embodiments of the invention can be implemented within compilers, for example. As is well known, a compiler generates machine executable object code from high-level source code, written in languages such as C++.

The foregoing describes only some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention, the embodiments being illustrative and not restrictive. For example, the equivalence of more than two sets of simultaneous linear algebraic equations may be determined by pair-wise comparing the sets for equivalence.